1. Hello World with Class

Write a Ruby script that prints “Hello, World!” using a class.

- Class `HelloWorld`
  - Use method that sets an instance
    - create variable called `message` that hold the string `Hello World!`
  - Create a method that display the message called `print_hello`

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ cat 1-main.rb
require_relative '1-hello_world_class'

# Create an instance of HelloWorld, change the message, and call the
print_hello method
hello_world_instance = HelloWorld.new

hello_world_instance.print_hello
```

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 1-main.rb
Hello, World!
```

Step 1: Create the `1-hello_world_class.rb` file

This file contains the class `HelloWorld`, which defines the `message` instance variable and a method `print_hello` to print the message.

```ruby
# 1-hello_world_class.rb

class HelloWorld
  def initialize
    @message = "Hello, World!"  # Set the message instance variable
  end

  # Method to print the message
  def print_hello
    puts @message
```

```
  end
end
```

Step 2: Create the `1-main.rb` file

This is the main script that requires the class file and calls the `print_hello` method.

```
# 1-main.rb

require_relative '1-hello_world_class'

# Create an instance of HelloWorld and call the print_hello method
hello_world_instance = HelloWorld.new
hello_world_instance.print_hello
```

Explanation:

1. **`HelloWorld` Class**: Defines the structure of the object with a `message` instance variable.
2. **`initialize` Method**: Called automatically when a new instance of `HelloWorld` is created. It sets the `@message` instance variable to `"Hello, World!"`.
3. **`print_hello` Method**: Prints the message stored in the `@message` instance variable when called.

This is a typical way to use classes in Ruby to organize and reuse functionality.